

MMM		MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA		AAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM		MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA		AAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM		MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA		AAAAAAAAA		CCCCCCCCCCCC	PPPPPPPPPPPP	
MMMMMM	MMMMMM		TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM		TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM		TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMM	MMM	MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMM	MMM	MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMM	MMM	MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	PPP
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM		MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM		MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM		MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM		MMM	TTT	AAA	AAA	AAA	AAA	CCCCCCCCCCCC	PPP	

[illegible]

```
0001 0
0002 0 MODULE ACCESS (LANGUAGE (BLISS32) ,
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1 ++
0031 1
0032 1 FACILITY: MTAACP
0033 1
0034 1 ABSTRACT:
0035 1 This module performs the access function for mtaacp.
0036 1
0037 1 ENVIRONMENT:
0038 1
0039 1 Starlet operating system, including privileged system services
0040 1 and internal exec routines.
0041 1
0042 1 --
0043 1
0044 1
0045 1
0046 1 AUTHOR: D. H. Gillespie, CREATION DATE: 13-MAY-77 16:20
0047 1
0048 1 MODIFIED BY:
0049 1
0050 1 V03-002 MMD0275 Meg Dumont, 23-Mar-1984 9:49
0051 1 Change order of calling the routines CHECK_ACCESS and
0052 1 CHECK_FILE_ACC
0053 1
0054 1 V03-001 MMD0001 Meg Dumont, 5-Nov-1982 15:53
0055 1 Added support for setting user handling of EOT on a file access
0056 1
0057 1 V02-005 DMW00050 David Michael Walp 10-Nov-1981
```



```
58      0058 1 | Return 'NO SUCH FILE' error for when a 'NO MORE FILES' error
59      0059 1 | is returned and no files have been found.
60      0060 1 |
61      0061 1 | V02-004 REFORMAT      Maria del C. Nasr      30-Jun-1980
62      0062 1 |
63      0063 1 | V02-003 SPR27361      Maria del C. Nasr      10-Jun-1980
64      0064 1 | Fix design problem in which the IO_PACKET was being returned
65      0065 1 | before IO was really completed. Now, a file will not be accessed
66      0066 1 | until the tape mark is successfully skipped. Also, the user
67      0067 1 | labels code was eliminated, since it was never used.
68      0068 1 |
69      0069 1 | A0002 MCN0001      Maria del C. Nasr      11-Sep-79  17:02
70      0070 1 | Added argument to CHECK_FILE_ACC call to fix bug in
71      0071 1 | "create if" function.
72      0072 1 |
73      0073 1 | **
74      0074 1 |
75      0075 1 | LIBRARY 'SYSS$LIBRARY:LIB.L32';
76      0076 1 |
77      0077 1 | REQUIRE 'SRCS:MTADEF.B32';
78      0461 1 |
79      0462 1 | FORWARD ROUTINE
80      0463 1 |     MTA_ACCESS : NOPRES NOVALUE,      ! main control for access function
81      0464 1 |     CHECK_FJND : COMMON_CALL,        ! conditional directory search
82      0465 1 |     HANDLER;      ! conditional handler to catch error exit
83      0466 1 |
```

```
85 0467 1 GLOBAL ROUTINE MTA_ACCESS : NOPRES NOVALUE =
86 0468 1
87 0469 1 ++
88 0470 1
89 0471 1 FUNCTIONAL DESCRIPTION:
90 0472 1 This is the main processing routine for mtaacp access function
91 0473 1
92 0474 1 CALLING SEQUENCE:
93 0475 1 MTA_ACCESS()
94 0476 1
95 0477 1 INPUT PARAMETERS:
96 0478 1 None
97 0479 1
98 0480 1 IMPLICIT INPUTS:
99 0481 1 CURRENT_UCB - address of current ucb
100 0482 1 CURRENT_WCB - address of current wcb
101 0483 1 CURRENT_VCB - address of current vcb
102 0484 1 IO_PACKET - address of current io request packet
103 0485 1
104 0486 1 OUTPUT PARAMETERS:
105 0487 1 None
106 0488 1
107 0489 1 IMPLICIT OUTPUTS:
108 0490 1 None
109 0491 1
110 0492 1 ROUTINE VALUE:
111 0493 1 None
112 0494 1
113 0495 1 SIDE EFFECTS:
114 0496 1 Access request processed
115 0497 1 USER ERRORS
116 0498 1 $$$_FILALRACC - file already accessed
117 0499 1 $$$_BADPARAM - bad input parameters
118 0500 1
119 0501 1 --
120 0502 1
121 0503 2 BEGIN
122 0504 2
123 0505 2 EXTERNAL REGISTER
124 0506 2 COMMON_REG;
125 0507 2
126 0508 2 LOCAL
127 0509 2 FUNCTION : BLOCK [1], ! function and qualifiers
128 0510 2 PACKET : REF BBLOCK, ! address of io packet
129 0511 2
130 0512 2 ! address of buffer descriptors
131 0513 2 !
132 0514 2 ABD : REF BBLOCKVECTOR [, ABD$C_LENGTH],
133 0515 2 FIB : REF BBLOCK; ! address of file information block
134 0516 2
135 0517 2 EXTERNAL
136 0518 2 CURRENT_UCB : REF BBLOCK, ! address of current ucb
137 0519 2 CURRENT_WCB : REF BBLOCK, ! address of window control block
138 0520 2 IO_PACKET : REF BBLOCK, ! address of io request packet
139 0521 2 USER_STATUS : VECTOR; ! status to return to user
140 0522 2
141 0523 2 EXTERNAL ROUTINE
```

```
142 0524 2
143 0525 2
144 0526 2
145 0527 2
146 0528 2
147 0529 2
148 0530 2
149 0531 2
150 0532 2
151 0533 2
152 0534 2
153 0535 2
154 0536 2
155 0537 2
156 0538 2
157 0539 2
158 0540 2
159 0541 2
160 0542 2
161 0543 2
162 0544 2
163 0545 2
164 0546 2
165 0547 2
166 0548 2
167 0549 2
168 0550 2
169 0551 2
170 0552 2
171 0553 2
172 0554 2
173 0555 2
174 0556 2
175 0557 2
176 0558 2
177 0559 2
178 0560 2
179 0561 2
180 0562 2
181 0563 2
182 0564 2
183 0565 2
184 0566 2
185 0567 2
186 0568 2
187 0569 2
188 0570 2
189 0571 2
190 0572 2
191 0573 2
192 0574 2
193 0575 2
194 0576 2
195 0577 2
196 0578 2
197 0579 2
198 0580 2

! Make data base changes nec for access
ACCESS_FILE : COMMON_CALL,
ACCESS_NEW_FILE : COMMON_CALL, ! access newly created file

! Check volume protection on each file access
CHECK_ACCESS : COMMON_CALL,
CHECK_FILE_ACC : COMMON_CALL, ! check access to the file
CLOSE_FILE : L$CLOSE_FILE, ! close out file
GET_FIB : COMMON_CALL, ! get file information block
MTA_CREATE, ! Main control for create function
POSITION BY FID : COMMON_CALL, ! position to file by fid
READ_ATTRIBUTE : COMMON_CALL, ! read file attributes
SET_USER_EOT : COMMON_CALL, ! call to set user eot handling
SPACE_TM : COMMON_CALL NOVALUE, ! space tape mark
START_VIO : COMMON_CALL; ! start up virtual io

! Setup pointers
PACKET = .IO_PACKET; ! address of io packet

! Address of buffer descriptors
ABD = .BBLOCK[.PACKET[IRP$L_SVAPTE], AIB$L_DESCRIPTOR];
FIB = GET_FIB(.ABD); ! get file information block

! Get the function to be performed
FUNCTION = .PACKET[IRP$W_FUNC];

! If anyone has a file accessed then another file can not be access nor can
! the tape be searched

IF .CURRENT_WCB NEQ 0
AND
(.FUNCTION[IO$V_ACCESS]
OR
.FIB[FIB$W_DID_NUM] NEQ 0)
THEN
ERR_EXIT(SS$_FILALRACC);

IF .FIB[FIB$V_TRUNC]
OR
.FUNCTION[IO$V_DELETE]
THEN
ERR_EXIT(SS$_BADPARAM);

! It is possible that a create was issued without an access in which case
! there is a partial file on this tape. if the access is to a different
! file then close out the partial file before preceding

IF .CURRENT_VCB[VCB$V_PARTFILE] ! if there is a partial file
THEN
```



```
BEGIN
! Is this a find or a request for a different file
!
IF .FIB[FIB$W_DID_NUM] NEQ 0
OR
.(FIB[FIB$W_FID])<0, 32> NEQU .CURRENT_VCB[VCB$S_CUR_FID]
THEN
CLOSE_FILE() ! close partial file before preceding
ELSE
BEGIN
! if any attributes requested,
!
IF .PACKET[IRP$W_BCNT] GTR ABD$C_ATTRIB
THEN
READ_ATTRIBUTE(.ABD); ! then read them
IF .FUNCTION[IO$V_ACCESS]
THEN
BEGIN
IF NOT .FIB[FIB$V_WRITE]
THEN
ERR_EXIT(SS$_BADPARAM); ! must write new file
ACCESS_NEW_FILE(.FIB, .PACKET, .ABD); ! access partial file
END;
RETURN;
END;
END;
! If file is accessed, the only function that can be performed is read
! attributes
!
IF .CURRENT_WCB NEQ 0
THEN
BEGIN
IF .PACKET[IRP$W_BCNT] GTR ABD$C_ATTRIB
THEN
READ_ATTRIBUTE(.ABD);
RETURN;
END;
END;
! If directory id given, find file by file name string
!
IF .FIB[FIB$W_DID_NUM] NEQ 0
```

```

: 256 0638 2
: 257 0639 2
: 258 0640 2
: 259 0641 2
: 260 0642 2
: 261 0643 2
: 262 0644 2
: 263 0645 2
: 264 0646 2
: 265 0647 2
: 266 0648 2
: 267 0649 2
: 268 0650 2
: 269 0651 2
: 270 0652 2
: 271 0653 2
: 272 0654 2
: 273 0655 2
: 274 0656 2
: 275 0657 2
: 276 0658 2
: 277 0659 2
: 278 0660 2
: 279 0661 2
: 280 0662 2
: 281 0663 2
: 282 0664 2
: 283 0665 2
: 284 0666 2
: 285 0667 2
: 286 0668 2
: 287 0669 2
: 288 0670 2
: 289 0671 2
: 290 0672 2
: 291 0673 2
: 292 0674 2
: 293 P 0675 2
: 294 0676 2
: 295 0677 2
: 296 0678 2
: 297 0679 2
: 298 0680 2
: 299 0681 2
: 300 0682 1

THEN
    IF NOT CHECK_FIND()
    THEN
        BEGIN
            IF NOT .FUNCTION[IOSV_CREATE]
            THEN
                ERR_EXIT(SS$_NOSUCHFILE);

            USER_STATUS[0] = SS$_CREATED;
            RETURN MTA_CREATE();          ! execute create function

        END;

    ! position volume set by fid
    POSITION_BY_FID(.(FIB[FIB$W_FID])<0, 32>, .FIB[FIB$W_FID_RVN]);

    IF .PACKET[IRP$W_BCNT] GTR ABD$C_ATTRIB
    THEN
        READ_ATTRIBUTE(.ABD);

    ! if access requested
    !
    IF .FUNCTION[IOSV_ACCESS]
    THEN
        BEGIN
            CHECK_FILE_ACC(1);          ! check access to file, (1)=mta_access
            CHECK_ACCESS(.FIB[FIB$V_WRITE]); ! check volume access

            IF .CURRENT_VCB[VCB$B_TM] EQLU 0
            THEN
                SPACE_TM(1);          ! position to data

            KERNEL_CALL(SET_USER_EOT, .FIB);          ! call to set user EOT handling
            KERNEL_CALL(ACCESS_FILE, .FIB[FIB$L_ACCTL], .PACKET[IRP$L_PID], 1,
                .FIB[FIB$V_WRITE], .ABD);
            KERNEL_CALL(START_VIO);          ! start virtual io
            RETURN;

        END;

    END;
END;
```

```

.TITLE ACCESS
.IDENT \V04-000\

.EXTRN CURRENT_UCB, CURRENT_WCB
.EXTRN IO_PACKET, USER_STATUS
.EXTRN ACCESS_FILE, ACCESS_NEW_FILE
.EXTRN CHECK_ACCESS, CHECK_FILE_ACC
.EXTRN CLOSE_FILE, GET_FIB
.EXTRN MTA_CREATE, POSITION_BY_FID
.EXTRN READ_ATTRIBUTE, SET_USER_EOT
```


				.EXTRN	SPACE TM, START_VIO		
				.EXTRN	SYSS\$CMKRNL		
				.PSECT	\$CODE\$,NOWRT,2		
			0000	00000	.ENTRY	MTA ACCESS, Save nothing	0467
	53	0000G	CF	D0 00002	MOVL	10 PACKET, PACKET	0545
	55	2C	B3	D0 00007	MOVL	34(PACKET), ABD	0549
			55	DD 0000B	PUSHL	ABD	0550
0000G	CF		01	FB 0000D	CALLS	#1, GET_FIB	
	52		50	D0 00012	MOVL	R0, FIB	
	54	20	A3	3C 00015	MOVZWL	32(PACKET), FUNCTION	0554
		0000G	CF	D5 00019	TSTL	CURRENT_WCB	0560
			0D	13 0001D	BEQL	2\$	
05	54		06	E0 0001F	BBS	#6, FUNCTION, 1\$	0562
		0A	A2	B5 00023	TSTW	10(FIB)	0564
			04	13 00026	BEQL	2\$	
		00A4	8F	BF 00028	CHMU	#164	0566
	04	17	A2	E8 0002C	BLBS	23(FIB), 3\$	0568
02	54		08	E1 00030	BBC	#8, FUNCTION, 4\$	0570
			14	BF 00034	CHMU	#20	0572
	30	0B	AB	E9 00036	BLBC	11(CURRENT_VCB), 9\$	0579
		0A	A2	B5 0003A	TSTW	10(FIB)	0586
			07	12 0003D	BNEQ	5\$	
24	AB	04	A2	D1 0003F	CMPL	4(FIB), 36(CURRENT_VCB)	0588
			05	13 00044	BEQL	6\$	
		0000G	30	00046	BSBW	CLOSE_FILE	0590
			1F	11 00049	BRB	9\$	
	05	32	A3	B1 0004B	CMPW	50(PACKET), #5	0597
			07	1B 0004F	BLEQU	7\$	
			55	DD 00051	PUSHL	ABD	0599
0000G	CF		01	FB 00053	CALLS	#1, READ_ATTRIBUTE	
	54		06	E1 00058	BBC	#6, FUNCTION, 14\$	0601
	02	01	A2	E8 0005C	BLBS	1(FIB), 8\$	0605
			14	BF 00060	CHMU	#20	0607
			2C	BB 00062	PUSHR	#*M<R2,R3,R5>	0609
0000G	CF		03	FB 00064	CALLS	#3, ACCESS_NEW_FILE	
			04	00069	RET		0592
		0000G	CF	D5 0006A	TSTL	CURRENT_WCB	0622
			0F	13 0006E	BEQL	11\$	
	05	32	A3	B1 00070	CMPW	50(PACKET), #5	0626
			01	1A 00074	BGTRU	10\$	
			04	00076	RET		
			55	DD 00077	PUSHL	ABD	0628
0000G	CF		01	FB 00079	CALLS	#1, READ_ATTRIBUTE	
			04	0007E	RET		0624
		0A	A2	B5 0007F	TSTW	10(FIB)	0637
			1D	13 00082	BEQL	13\$	
0000V	CF		00	FB 00084	CALLS	#0, CHECK_FIND	0640
	15		50	E8 00089	BLBS	R0, 13\$	
			54	95 0008C	TSTB	FUNCTION	0644
			04	19 0008E	BLSS	12\$	
		0910	8F	BF 00090	CHMU	#2320	0646
0000G	CF	0619	8F	3C 00094	MOVZWL	#1561, USER STATUS	0648
0000G	CF		00	FB 0009B	CALLS	#0, MTA_CREATE	0649
			04	000A0	RET		
	7E	08	A2	3C 000A1	MOVZWL	8(FIB), -(SP)	0655

			0000G	CF	04	A2	DD	000A5	PUSHL	4(FIB)		
				05	32	02	FB	000A8	CALLS	#2, POSITION BY_FID		
						A3	B1	000AD	CMPL	50(PACKET), #5		0657
						07	1B	000B1	BLEQU	14\$		
			0000G	CF		55	DD	000B3	PUSHL	ABD		0659
		5C		54		01	FB	000B5	CALLS	#1, READ ATTRIBUTE		
						06	E1	000BA	BBC	#6, FUNCTION, 16\$		0664
			0000G	CF		01	DD	000BE	PUSHL	#1		0667
7E	01	A2		01		01	FB	000C0	CALLS	#1, CHECK_FILE_ACC		
			0000G	CF		00	EF	000C5	EXTZV	#0, #1, 1(FIB), -(SP)		0668
					2E	01	FB	000CB	CALLS	#1, CHECK_ACCESS		
						AB	95	000D0	TSTB	46(CURRENT_VCB)		0670
						07	12	000D3	BNEQ	15\$		
			0000G	CF		01	DD	000D5	PUSHL	#1		0672
						01	FB	000D7	CALLS	#1, SPACE_TM		
						52	DD	000DC	PUSHL	FIB		0674
						01	DD	000DE	PUSHL	#1		
					0000G	5E	DD	000E0	PUSHL	SP		
			00000000G	9F		CF	9F	000E2	PUSHAB	SET_USER EOT		
						04	FB	000E6	CALLS	#4, @#SYSS\$CMKRNL		
						55	DD	000ED	PUSHL	ABD		0676
7E	01	A2		01		00	EF	000EF	EXTZV	#0, #1, 1(FIB), -(SP)		
					0C	01	DD	000F5	PUSHL	#1		
						A3	DD	000F7	PUSHL	12(PACKET)		
						62	DD	000FA	PUSHL	(FIB)		
						05	DD	000FC	PUSHL	#5		
					0000G	5E	DD	000FE	PUSHL	SP		
			00000000G	9F		CF	9F	00100	PUSHAB	ACCESS FILE		
						08	FB	00104	CALLS	#8, @#SYSS\$CMKRNL		
						7E	D4	0010B	CLRL	-(SP)		0677
					0000G	5E	DD	0010D	PUSHL	SP		
			00000000G	9F		CF	9F	0010F	PUSHAB	START VIO		
						03	FB	00113	CALLS	#3, @#SYSS\$CMKRNL		
						04	0011A	16\$:	RET			0682

; Routine Size: 283 bytes, Routine Base: \$CODE\$ + 0000

; 301 0683 1

.EXTRN FIND_FILE

				0000 00000 CHECK_FIND:			
	6D	0000V	CF	9E 00002	.WORD	Save nothing	: 0684
0000G	CF		00	FB 00007	MOVAB	HANDLER, (FP)	: 0735
	50		01	D0 0000C	CALLS	#0, FIND_FILE	: 0736
				04 0000F	MOVL	#1, R0	: 0737
					RET		: 0739

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 011B


```
0740 1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
0741 1
0742 1 ++
0743 1
0744 1 FUNCTIONAL DESCRIPTION:
0745 1
0746 1 This routine is the condition handler for the conditional find call.
0747 1 It intercepts the error exit from FIND and unwinds to CHECK_FIND's
0748 1 caller when appropriate.
0749 1
0750 1
0751 1 CALLING SEQUENCE:
0752 1 HANDLER (ARG1, ARG2)
0753 1
0754 1 INPUT PARAMETERS:
0755 1 ARG1: address of signal array
0756 1 ARG2: address of mechanism array
0757 1
0758 1 IMPLICIT INPUTS:
0759 1 None
0760 1
0761 1 OUTPUT PARAMETERS:
0762 1 None
0763 1
0764 1 IMPLICIT OUTPUTS:
0765 1 None
0766 1
0767 1 ROUTINE VALUE:
0768 1 SS$_RESIGNAL or none if unwind
0769 1
0770 1 SIDE EFFECTS:
0771 1 None
0772 1
0773 1 --
0774 1
0775 2 BEGIN
0776 2
0777 2 MAP
0778 2 SIGNAL : REF BBLOCK, ! signal arg array
0779 2 MECHANISM : REF BBLOCK; ! mechanism arg array
0780 2
0781 2 EXTERNAL ROUTINE
0782 2 SYSSUNWIND : ADDRESSING_MODE (ABSOLUTE); ! system unwind service
0783 2
0784 2 EXTERNAL
0785 2 LOCAL_FIB : BBLOCK; ! copy of user's fib
0786 2
0787 2 ! If the condition is change mode to user (error exit) and the status is
0788 2 ! no such file, cause an unwind to return 0 to the access main line.
0789 2 ! Otherwise, just resignal the condition.
0790 2
0791 2
0792 2 IF .SIGNAL[CHF$_SIG_NAME] EQL SS$_CMODUSER
0793 2 AND
0794 2 .SIGNAL[CHF$_SIG_ARG1] EQL SS$_NOSUCHFILE
0795 2 THEN
0796 3 BEGIN
```

```
417 0797 3
418 0798 3
419 0799 3
420 0800 4
421 0801 4
422 0802 4
423 0803 5
424 0804 5
425 0805 5
426 0806 5
427 0807 5
428 0808 5
429 0809 4
430 0810 4
431 0811 3
432 0812 4
433 0813 4
434 0814 4
435 0815 4
436 0816 4
437 0817 4
438 0818 4
439 0819 4
440 0820 3
441 0821 3
442 0822 2
443 0823 2
444 0824 2
445 0825 2
446 0826 1

IF .LOCAL_FIB[FIB$V_WILD]
THEN
  BEGIN
    IF .LOCAL_FIB[FIB$L_WCC] NEQ 0
    THEN
      BEGIN
        SIGNAL[CHF$L_SIG_ARG1] = SS$_NOMOREFILES;
        ! clear context if at end of search
        LOCAL_FIB[FIB$L_WCC] = 0;
      END;
    END
  ELSE
    BEGIN
      ! clear context if at end of search
      LOCAL_FIB[FIB$L_WCC] = 0;
      MECHANISM[CHF$L_MCH_SAVRO] = 0;
      SYSSUNWIND(0, 0);
    END;
  END;
RETURN SS$_RESIGNAL;
END;

! status is irrelevant if unwinding
! end of routine HANDLER
```

```
.EXTRN SYSSUNWIND, LOCAL_FIB

0004 00000 HANDLER: .WORD Save R2
52 0000G CF 9E 00002 MOVAB LOCAL_FIB+16, R2
50 04 AC D0 00007 MOVL SIGNAL, R0
00000424 8F 04 A0 D1 0000B CMPL 4(R0), #1060
2E 12 00013 BNEQ 2$
00000910 8F 08 A0 D1 00015 CMPL 8(R0), #2320
24 12 0001D BNEQ 2$
0E 05 A2 E9 0001F BLBC LOCAL_FIB+21, 1$
62 D5 00023 TSTL LOCAL_FIB+16
1C 13 00025 BEQL 2$
08 A0 0930 8F 3C 00027 MOVZWL #2352, 8(R0)
62 D4 0002D CLRL LOCAL_FIB+16
12 11 0002F BRB 2$
50 08 AC D0 00031 1$: CLRL LOCAL_FIB+16
OC A0 D4 00033 MOVL MECHANISM, R0
7E 7C 0003A CLRL 12(R0)
00000000G 9F 02 FB 0003A CLRQ -(SP)
50 0918 8F 3C 00043 2$: CALLS #2, @#SYSSUNWIND
04 00048 RET MOVZWL #2328, R0
```

0740
0792
0794
0798
0801
0804
0808
0798
0816
0818
0819
0824
0826

; Routine Size: 73 bytes, Routine Base: \$CODE\$ + 012B


```
: 447      0827 1 END
: 448      0828 1
: 449      0829 0 ELUDOM
```

PSECT SUMMARY

```
:
: Name          Bytes          Attributes
: $CODE$        372 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
```

Library Statistics

```
:
: File          ----- Symbols ----- Pages Processing
:               Total   Loaded   Percent   Mapped   Time
: _$255$DUA28:[SYSLIB]LIB.L32;1  18619      35        0      1000     00:01.9
```

COMMAND QUALIFIERS

```
:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:ACCESS/OBJ=OBJ$:ACCESS MSRC$:ACCESS/UPDATE=(ENH$:ACCESS)
```

```
: Size:          372 code + 0 data bytes
: Run Time:      00:11.9
: Elapsed Time:  00:43.0
: Lines/CPU Min: 4179
: Lexemes/CPU-Min: 19709
: Memory Used:   142 pages
: Compilation Complete
```


0253

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY